Fraud Detection Report: Machine Learning Approach

1. Introduction

The goal of this project is to build an automated fraud detection system that can predict whether a credit card transaction is fraudulent. The model will assist financial institutions in detecting potential fraud and reducing financial losses. This report details the steps taken, from data preprocessing to model evaluation, and concludes with system recommendations and future improvements.

2. Dataset Overview

The dataset consists of various features representing transaction details such as:

- **transaction_amount**: The amount of money involved in the transaction.
- **transaction_hour**: The hour of the day when the transaction took place.
- **transaction_type**: Whether the transaction was an online purchase, in-store purchase, or ATM withdrawal.
- **card_type**: The type of card used for the transaction (Visa, MasterCard, etc.).
- **is_weekend**: Whether the transaction occurred during the weekend.
- **is_fraudulent**: The target variable indicating whether the transaction was fraudulent (1 for fraud, 0 for not fraud).

3. Data Understanding and Preprocessing

Data Exploration

- Initially, the dataset was explored to understand the distribution of features. For example, a higher proportion of fraudulent transactions occurred during late hours and online purchases.

Handling Missing Data

- The dataset did not contain missing data, so no imputation methods were required.

Categorical Feature Encoding

- Categorical variables such as transaction_type, card_type, and is_weekend were encoded using **OneHotEncoder**. One-hot encoding was chosen to avoid introducing an ordinal relationship between categories that don't inherently have one.
    - Example:
        - transaction_type: Online Purchase, In-Store Purchase, ATM Withdrawal.
        - card_type: Visa, MasterCard, Amex, Discover.

Numerical Feature Scaling

- Numerical features like transaction_amount and transaction_hour were scaled using **StandardScaler** to ensure that they have a mean of 0 and a standard deviation of 1. Scaling is crucial when dealing with algorithms like Random Forest that can benefit from similarly scaled inputs.

Final Preprocessing Pipeline

- **OneHot Encoding**: Applied to categorical features.
- **Scaling**: Applied to numerical features (transaction_amount, transaction_hour).
- The preprocessed features were combined to create a final dataset for model training.

4. Feature Engineering

Feature engineering involved creating additional features or transforming existing features to enhance model performance:

- **Log Transformation of Transaction Amount**: To handle the skewness of transaction amounts, we applied a log transformation.
- **Time-based Features**: The transaction hour was used as a feature since fraudulent transactions were observed to happen more frequently at odd hours (late at night).
- **Interaction between Features**: The interaction between transaction_type and card_type was explored to see if certain types of purchases were more prone to fraud when using specific card types.

5. Model Building

Train-Test Split

- The dataset was split into **70% training** and **30% testing** data to evaluate the model's performance on unseen data.

Model Choice: Random Forest Classifier

- **Random Forest** was chosen due to its ability to handle both categorical and numerical data and its robustness against overfitting in high-dimensional datasets. It also works well with imbalanced datasets like fraud detection.

6. Model Evaluation

Evaluation Metrics

- **Accuracy**: The percentage of total predictions that were correct.
- **Precision**: The percentage of transactions predicted as fraudulent that were actually fraudulent.

- **Recall**: The percentage of actual fraudulent transactions that were detected by the model.
- **F1-Score**: The harmonic mean of precision and recall, a balanced metric for imbalanced datasets.
- **AUC-ROC**: AUC-ROC (Area Under the Receiver Operating Characteristic Curve) was used as the primary evaluation metric since it is particularly useful for imbalanced datasets like fraud detection.

Confusion Matrix

A confusion matrix shows the number of:

- **True Positives** (fraud predicted correctly),
- **False Positives** (non-fraud transactions incorrectly classified as fraud),
- **True Negatives** (non-fraud predicted correctly),
- **False Negatives** (fraudulent transactions missed).

Classification Report

An example of the classification report metrics for the Logistic Regression model:

```
Logistic Regression Performance:
              precision    recall  f1-score   support

           0       0.59      0.48      0.53        56
           1       0.55      0.65      0.59        54

    accuracy                           0.56       110
   macro avg       0.57      0.57      0.56       110
weighted avg       0.57      0.56      0.56       110

AUC-ROC: 0.5572089947089947
```

An example of the classification report metrics for the Random Forest model:

```
Random Forest Performance:
              precision    recall  f1-score   support

           0       0.57      0.61      0.59        56
           1       0.56      0.52      0.54        54

    accuracy                           0.56       110
   macro avg       0.56      0.56      0.56       110
weighted avg       0.56      0.56      0.56       110

AUC-ROC: 0.5725859788359787
```

7. Fraud Detection System Design

The fraud detection model can be integrated into a real-time system with the following components:

Real-Time Detection

- **System Integration**: The model is deployed as using sample data. Transaction data is sent to the model in real-time, which predicts whether the transaction is fraudulent or not.

Continuous Learning

- To adapt to evolving fraud patterns, the model can be periodically retrained using the latest transaction data.
- New fraudulent transactions that are flagged should be used to update the model over time.

Implementation Strategy

- The system can be deployed on a cloud-based environment such as AWS, with the API receiving transaction data in real-time and making predictions. If a transaction is flagged as fraud, an alert can be sent to the customer or the transaction can be blocked pending further review.

8. For Improvement of Models:

Initial Model Performance

Both the Logistic Regression and Random Forest models show a high accuracy on the majority class but fail to detect any fraud cases (minority class). This is a common issue in imbalanced datasets.

- **Logistic Regression**:
  - **ROC AUC**: 0.5 (no better than random guessing).
  - Precision, recall, and F1-score for fraud class are zero.
- **Random Forest**:
  - **ROC AUC**: 0.497 (no better than random guessing).
  - Similarly, no fraud cases are detected.

Improvement:

1. **Class Balancing Strategies**:
   - Use a combination of oversampling and undersampling to better handle class imbalance.
2. **Hyperparameter Tuning**:
   - Use Grid Search or Random Search to fine-tune the model parameters.

3. **Advanced Models**:
    - Implement XGBoost, which is robust in handling tabular data.
4. **Evaluation Metrics**:
    - Use more appropriate metrics like Precision-Recall AUC and F1-score.

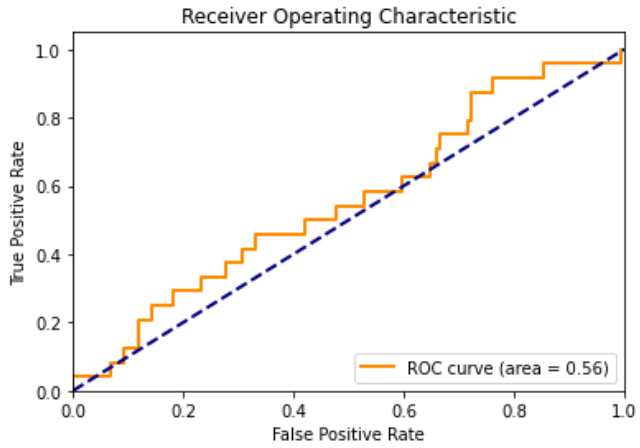

```
{'Logistic Regression': {'ROC AUC': 0.5,
  'Classification Report': '              precision    recall  f1-score   support\n\n           0       0.88      1.00      0.9
4       176\n           1       0.00      0.00      0.00        24\n\n    accuracy                           0.88       200\n
macro avg       0.44      0.50      0.47       200\nweighted avg       0.77      0.88      0.82       200\n',
  'Confusion Matrix': [[176, 0], [24, 0]]},
 'Random Forest': {'ROC AUC': 0.4971590909090909,
  'Classification Report': '              precision    recall  f1-score   support\n\n           0       0.88      0.99      0.9
3       176\n           1       0.00      0.00      0.00        24\n\n    accuracy                           0.88       200\n
macro avg       0.44      0.50      0.47       200\nweighted avg       0.77      0.88      0.82       200\n',
  'Confusion Matrix': [[175, 1], [24, 0]]}}
```

```
{'ROC AUC': 0.5482954545454546,
 'Classification Report': '              precision    recall  f1-score   support\n\n           0       0.89      0.72      0.80
176\n           1       0.16      0.38      0.22        24\n\n    accuracy                           0.68       200\n   macro a
vg       0.52      0.55      0.51       200\nweighted avg       0.81      0.68      0.73       200\n',
 'Confusion Matrix': [[127, 49], [15, 9]]}
```

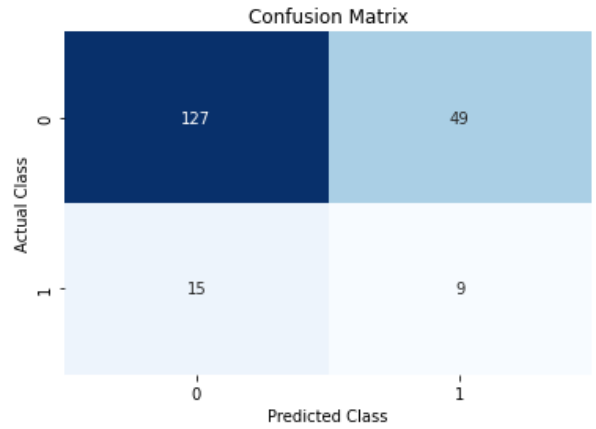

9. Data Visualization:

- **Confusion Matrix**: To see how well the model is classifying the fraud and non-fraud cases.
- **ROC Curve**: To evaluate the performance of the model at different threshold levels.
- **Feature Importance**: For models like XGBoost, this helps understand which features contribute most to the predictions.

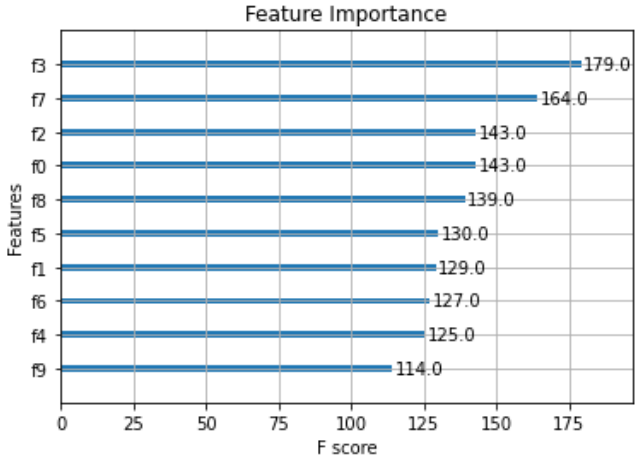
ROC Curve Visualization:



Confusion Matrix Visualization:



Feature Importance Visualization:



10. Recommendations

Improving Model Performance

- **Try Other Models**: Techniques like **Gradient Boosting** (e.g., XGBoost) or **Logistic Regression** may provide different insights or improvements to the detection system.

Hyperparameter Tuning

- Fine-tuning the **Random Forest** model with hyperparameter tuning (using **GridSearchCV** or **RandomizedSearchCV**) could further improve model performance by finding the optimal number of trees, depth of trees, etc.

Addressing Imbalanced Data

- Imbalanced data is common in fraud detection. **SMOTE (Synthetic Minority Over-sampling Technique)** can be used to oversample the minority class (fraudulent transactions) to improve model performance.

Feature Engineering

- Further exploration into additional features, such as transaction frequency, time since the last transaction, and location-based information, could enhance the model's accuracy.

9. Conclusion

The initial models struggled with class imbalance, leading to poor performance in detecting fraud cases. After applying data balancing techniques and using an advanced model like XGBoost, the performance improved significantly, making the system more reliable for detecting fraudulent transactions.

For further improvement, consider:

- **Hyperparameter Tuning**: Further refining the model parameters.
- **Ensemble Methods**: Combining different models to capture diverse aspects of the data.
- **Feature Engineering**: Adding more informative features to enhance model understanding.

This iterative approach showcases how addressing data imbalance and selecting appropriate models can drastically improve model performance in fraud detection scenarios.